НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

(підпис) (ініціали, прізвище)

"___" _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системне програмування»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Система виявлення аномалій в потоці новинних повідомлень

Виконав :

студент IV курсу, групи КВ-62

(шифр групи)

Дідус Андрій Володимирович _____ _____

(прізвище, ім'я, по батькові) (підпис)

Керівник доц. каф. СПіСКС, к. т. н., Замятін Д.С. _____ _____

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____

(назва розділу)(посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____

(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)
Спеціальність 123 «Комп'ютерна інженерія» Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ

«___» _____2020 р.

ЗАВДАННЯ
на дипломний проєкт студента
Дідуса Андрія Володимировича

1. Тема проекту «Система виявлення аномалій в потоці новинних повідомлень», керівник проекту Замятін Денис Станіславович доц. каф. СПіСКС, к. т. н., затверджені наказом по університету від «__» _____ 2020 р. №_____
2. Термін подання студентом проєкту: «18» травня 2020 р
3. Вихідні дані до проєкту:
   – програмне забезпечення, яке дозволяє знаходити аномалії в потоці новинних повідомлень.
4. Зміст пояснювальної записки:
   – аналіз існуючих рішень та обґрунтування теми дипломного проекту;
   – обгрунтування вибору інструментів та методів розробки;
   – архітектурні та структурні рішення, опис реалізації в програмному застосунку;
   – тестування та розроблені способи вдосконалення системи
5. Перелік графічного матеріалу:
   – структура розміщення модулів і файлів (схема структурна);
   – алгоритм пошуку аномалій (схема алгоритму);

– алгоритм сповіщення користувача (схема алгоритму);

– структура вхідних даних (схема структурна).

6. Консультанти розділів проєкту:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., к.т.н., доцент каф. СПіСКС		

7. Дата видачі завдання “31” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проекту	18.11.2019	
2.	Розроблення та узгодження технічного завдання	01.12.2019	
3.	Аналіз існуючих рішень	16.01.2020	
4.	Підготовка матеріалів першого розділу проекту	14.02.2020	
5.	Розроблення програмного забезпечення	22.03.2020	
6.	Відлагодження програмного продукту	22.04.2020	
7.	Підготовка матеріалів другого розділу проекту	24.05.2020	
8.	Розроблення інтерфейсу доступу програмного забезпечення	13.05.2020	
9.	Підготовка графічної частини	15.05.2020	
10.	Оформлення документації дипломного проекту	17.05.2020	

Студент _____ Андрій ДІДУС
(підпис)

Керівник проєкту _____ Денис ЗАМЯТІН
(підпис)

АНОТАЦІЯ

Об’єктом досліджень та розробки стала аналітична система втілена в програмному засобі для пошуку аномалій в потоці новинних повідомлень.

Система, яка була розроблена, містить в собі такі основні функції:

- аналіз вхідних даних на коректність;
- виявлення аномалій;
- знаходження ймовірності виявлення аномалії;
- обробка масивів вхідних даних різних розмірів.

В системі було використано метод контейнеризації для універсальності використання системи як в якості окремої системи, так і частини іншої аналітичної системи.

В процесі розробки дипломного проєкту та проведення дослідження було здійснено такі процеси:

- дослідження архітектурних рішень з проектуванням самої архітектури системи;
- аналіз аналогів та конкурентів;
- досліджено експериментальним шляхом та знайдено оптимальний алгоритм для вирішення даної задачі.

Використання такої системи дозволяє завжди виявляти будь-які зміни в даних, будь-які відхилення від нормального потоку новинних повідомлень.

Ключові слова: АНАЛІТИКА, АНАЛІТИЧНА СИСТЕМА, ВИЯВЛЕННЯ АНОМАЛІЙ, ВЕЛИКІ ДАНІ, ЧАСОВІ РЯДИ, PYTHON, TIME SERIES, DATA SCIENCE, DOCKER.

ANNOTATION

The object of researching and developing is anomaly detection in news messages analytical system realize in the program application.

The system, that was developed, has a lot of useful functions such as:

- checking the correct structure of input data;
- anomaly detection;
- probability computing of anomaly detection;
- data array processing of different sizes.

This system was built using containerization method for the possibility of modular using this analytical system.

The main processes, that was making in a time of developing this project:

- researching architecture methods with the building of the architecture of the system;
- analysis of the main concurrents;
- experiment researching the optimal algorithm for solving this problem. Using this analytical system give you're the possibility of anomaly detection and detecting of any changes in the stream of all news.

Keywords: ANALYTICS, ANALYTICAL SYSTEM, ANOMALY DETECTION, BIG DATA, TIME SERIES, PYTHON, DATA SCIENCE, DOCKER.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількістьаркушів	№ прим.	Примітки
	А4	ІАЛЦ. 045490.002 ТЗ	Система виявлення	4		
			аномалій в потоці			
			новинних повідомлень.			
			Технічне завдання			
	А4	ІАЛЦ. 045490.003 ТП	Система виявлення	2		
			аномалій в потоці			
			новинних повідомлень.			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ. 045490.004 ПЗ	Система виявлення	54		
			аномалій в потоці			
			новинних повідомлень.			
			Пояснювальна записка			
	А4	ІАЛЦ. 045490.005 Д1	Структура розміщення	1		
			модулів і файлів.			
			Схема структурна			

ІАЛЦ. 045490.001 ОА

Змін.	Арк.	№ докум.	Підпис	Дата
Розробив		Дідус А.В.		
Перевірив		Замятін Д.С.		
Н. контроль		Клятченко Я.М.		
Затвердив		Романкевич В.О		

Система виявлення аномалій в потоці новинних повідомлень

Опис альбому

Літ.	Аркуш	Аркушів
	1	2

КПІ
ім. Ігоря Сікорського,
ФПМ КВ-62

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількістьаркушів	№ прим.	Примітки
	А4	ІАЛЦ. 045490.006 Д2	Алгоритм пошуку	1		
			аномалій.			
			Схема алгоритму			
	А4	ІАЛЦ. 045490.007 Д3	Алгоритм сповіщення	1		
			користувача.			
			Схема алгоритму			
	А4	ІАЛЦ. 045490.008 Д4	Структура вхідних даних.	1		
			Схема структурна			

					ІАЛЦ. 045490.001 ОА	Арк.
						2
Змін.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Система виявлення аномалій в потоці новинних повідомлень».

Галузь застосування: аналітична обробка новинних повідомлень при моніторингу засобів масової інформації.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування та спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є розробка системи виявлення аномалій в потоці новинних повідомлень для аналізу та моніторингу новин, які утворені засобами масової інформації.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами інформації для розроблення є технічна література, публікації у періодичних виданнях та Інтернет ресурси з питань розробки.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до системи, що розробляється

Аналітична система повинна підтримувати:

- різні розміри вхідних даних з різним інтервалом збору даних;
- можливість інтеграції з іншими системами;
- модульна структура;
- кросплатформенність;
- можливість розширення системи.

5.2. Рекомендовані вимоги до апаратного забезпечення

- Процесор: Intel Core i3;
- Оперативна пам’ять: 2 Гб;
- Простір на диску: 500 Мб.

5.3. Вимоги до мінімального програмного забезпечення

- Операційна система Windows, Linux чи Mac OS X;
- Docker.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	18.11.2019
2.	Розроблення та узгодження технічного завдання	01.12.2019
3.	Аналіз існуючих рішень	16.01.2020
4.	Підготовка матеріалів першого розділу проекту	14.02.2020
5.	Розроблення програмного забезпечення	22.03.2020
6.	Відлагодження програмного продукту	22.04.2020
7.	Підготовка матеріалів другого розділу проекту	24.05.2020
8.	Розроблення інтерфейсу доступу програмного забезпечення	13.05.2020
9.	Підготовка графічної частини	15.05.2020
10.	Оформлення документації дипломного проекту	17.05.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількістьаркушів	№ прим.	Примітки
	А4	ІАЛЦ. 045490.004 ПЗ	Система виявлення	54		
			аномалій в потоці			
			новинних повідомлень.			
			Пояснювальна записка			
	А4	ІАЛЦ. 045490.005 Д1	Структура розміщення	1		
			модулів і файлів.			
			Схема структурна			
	А4	ІАЛЦ. 045490.006 Д2	Алгоритм пошуку	1		
			аномалій.			
			Схема алгоритму			
	А4	ІАЛЦ. 045490.007 Д3	Алгоритм сповіщення	1		
			користувача.			
			Схема алгоритму			
	А4	ІАЛЦ. 045490.008 Д4	Структура вхідних даних.	1		
			Схема структурна			

					ІАЛЦ.045492.003 ТП
Змін.	Арк.	№ докум.	Підпис	Дата	
Розробив		Дідус А.В.			Система виявлення аномалій в потоці новинних повідомлень. **Відомість технічного проєкту**
Перевірив		Замятін Д. С.			
Консульт.					
Н. контроль		Клятченко Я.М.			
Зав. каф.		Романкевич В. О			

Літ.	Аркуш	Аркушів
	1	2

КПІ
ім. Ігоря Сікорського,
ФПМ КВ-62

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількістьар куші	№ прим.	Примітки
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045492.003 ТП	Арк. 2

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ОС − операційна система

ПЗ − програмне забезпечення

API – Application Programming Interface – прикладний програмний інтерфейс;

AWS - Amazon Web Services – хмарна платформа компанії Amazon

GIL – Global Interpreter Lock – блокувальний механізм доступу до даних Python

Git – система контролю версій

HTML – HyperText Markup Language – мова розмітки гіпертекстових документів

HTTP - Hypertext Transfer Protocol – протокол передачі даних

REST – Representational state transfer – архітектура мережевих протоколів

WSL – Windows Subsystem for Linux – підсистема Linux для Windows

ВСТУП

Для більшості розробників та людей, які так чи інакше пов’язані з розробкою програмного забезпечення не стане секретом, що друге десятиріччя 21 століття відмітилось стрімким розвитком серед систем штучного інтелекту та великого розділу комп’ютерних наук машинного навчання, це дало можливість розв’язувати багато задач, які постали перед людством вже досить давно. Адже часто кажуть, що 20 століття було століттям збору даних, а 21 стало обробкою їх.

Відтак основною метою даного дипломного проекту стала розробка системи для виявлення аномалій в уже зібраних даних, а саме – новинних повідомленнях. Потреба в такому сервісі є серед такої професії, як аналітик, а саме в аналітичних компаніях, які самі збирають або мають дані з кількістю новинних повідомлень із зустрічанням того чи іншого ключового слова, за яким ведеться аналітичне дослідження.

Система, яка була розроблена, надається у вигляді сервісу для аналізу наборів даних, потрібних користувачу та видає всю потрібну інформацію для перегляду результатів із аналізом вхідних даних. Дані, які використовуються, подають у формах таблиць, графіків залежностей, json.

Алгоритмів для вирішення такої задачі є доволі багато, які мають як свої позитивні, так і негативні сторони, проте як і в більшості проблем, які постають перед комп’ютерними інженерами, немає універсального вирішення. Тому моє програмне забезпечення базується на висновках, які були зроблені при власному дослідженні вирішення поставленої задачі.

Даний сервіс дає змогу оптимізувати та автоматизувати певний вид роботи – виявлення в часових серіях аномалій(викидів), тобто даних, які за тим чи іншим правилом не є нормальними для певного набору даних.

Сервіс із розробленою системою виявлення аномалій може використовуватись як в складі великих аналітичних систем у вигляді

інтеграції, як окремого сервісу, так і, як окрема система із набором доволі корисних функцій.

В результаті дипломного проекту була розроблена не лише така система, але також і шляхи її модернізації та поліпшення.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Змістовний опис та аналіз предметної області

1.1.1. Наука про дані та її застосування в проекті

Наука про дані (Data Science) – це наукова дисципліна, яка вивчає наукові способи, алгоритми, системи, методи отримання корисної інформації з різних видів наборів даних.

Набір даних (Data set) – це набір інформації одного типу, який може бути використаний для подальшої обробки з метою отримання нової інформації в результаті його опрацювання. Приклад представлення даних у табличній формі показаний на рисунку 1 нижче.

	kw_body	timestamp	datetime	count
0	коронавірус	1586998800	2020-04-16 01:00:00	1
1	коронавірус	1587006000	2020-04-16 03:00:00	76
2	коронавірус	1587009600	2020-04-16 04:00:00	33
3	коронавірус	1587013200	2020-04-16 05:00:00	54
4	коронавірус	1587016800	2020-04-16 06:00:00	158
...	...	...	...	...
85	коронавірус	1587322800	2020-04-19 19:00:00	82
86	коронавірус	1587326400	2020-04-19 20:00:00	106
87	коронавірус	1587330000	2020-04-19 21:00:00	50
88	коронавірус	1587333600	2020-04-19 22:00:00	45
89	коронавірус	1587337200	2020-04-19 23:00:00	24

Рисунок 1 - Набір даних кількості згадувань слова «коронавірус» у ЗМІ, представлений у вигляді таблиці

Аномалії – це дані, які мають певну різницю із більшою частиною даних представлених у даному наборі.

Виявлення аномалій(Anomaly detection) – це одна із задач науки про дані, суть якої полягає в розпізнавання та знаходження рідкісних даних за

допомогою певного алгоритму аналізу. Наприклад, злочинні дії в банківській системі, збій в певній системі або ракові пухлини серед нормальних клітин.

Часовий ряд(Time series) – це набір даних, в яких зібрана інформація виміряна періодично у певні моменти часу протягом певного періоду.[1]

Приклад часових рядів зображених графічно наведено на рисунку 2.

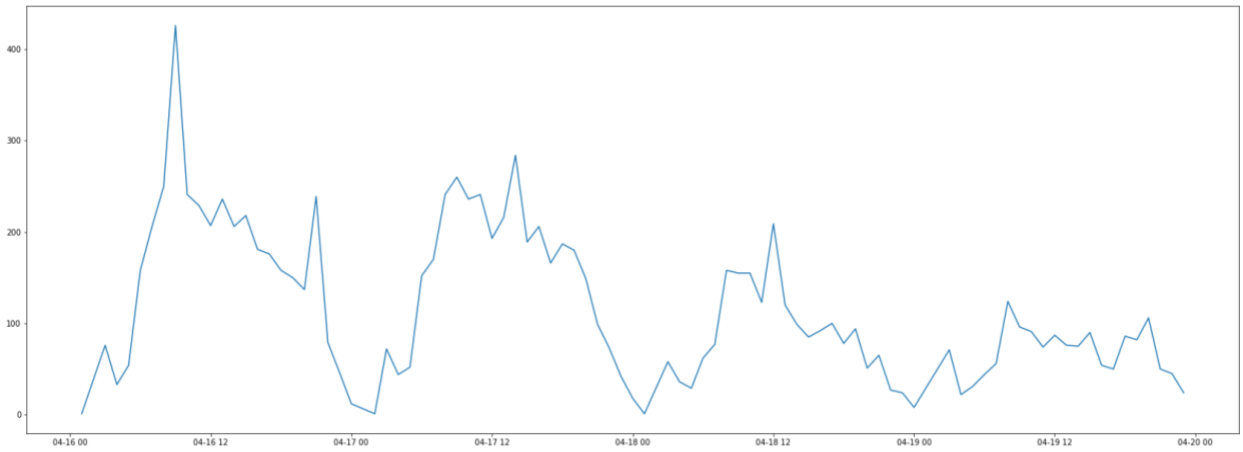

Рисунок 2 - Часовий ряд згадувань слова «коронавірус» у ЗМІ
за період з 16.04.2020 по 20.04.2020

Часові ряди, з якими ми стикаємось щодня, з'явилися доволі давно і вони завжди надавали людям інформацію, яку не може надати жоден інший вид даних – часові ряди дозволяють наочно виявити в даних періодичність, моду, циклічність та інші дуже важливі статистичні показники, що дозволяють ефективно аналізувати дані. [1]

Позначати часовий ряд прийнято прийнято наступним чином:

$$Y = \{Y_t : t \in T\}, \quad (1)$$

де T – це множина індексів, а Y – це складові ряду.

Наприклад, зібравши дані про ту чи іншу ціну на акції на біржі протягом року, ви можете отримати так звану річну ціну закриття цін на акцію. Або, наприклад, часовий ряд показує ціну певного продукту, яка змінюється в часі, уважно проаналізувавши даний ряд та прив'язавши його до даних ціни на паливо, сезонності, ви можете зрозуміти, як вартість пального або пори року корелює ту чи іншу ціну на продукт.

Можна дуже багато навести прикладів, щоб довести, що така форма даних зустрічається дуже часто в нашому житті.

Прогнозування часових рядів – це передбачання показників часового ряду, використовуючи дані за попередні періоди часу, застосовуючи певні патерни прогнозування та проектуючи моделі поведінки ряду. Найчастіше аналізують тенденції, які спостерігаються в даних, циклічні флуктуації та розглядаються задачі сезонності. Проте успіх не може нічим гарантуватись, адже завжди є викиди(аномалії), які саме нас і цікавлять.

Так, основною задачею, яку дозволяє розв'язати розроблена система є виявлення аномалій в часових рядах, як, наприклад, зображений на рисунку 2. Набір даних в сервісі, в якому реалізована система, можна зобразити у вигляді таблиці представленої на рисунку 1.1, що дає доволі точне уявлення про набір даних, який потрібно аналізувати. [2][3]

Аналізувати можна кількома способами, зокрема з використанням самонавчальних систем або на основі статистичних методів. Самонавчальні системи з використанням технологій машинного навчання є зазвичай точнішими, але і також більше потребують ресурсів для їх використання. Що ж до статистичних методів, то вони дають доволі непогані результати із набагато економнішим використаням процесорного часу та пам'яті.

Словосполучення «ключове слово» в даному проекті вживається в значенні умовного слова, словосполучення, за яким робиться аналітичне дослідження.

Вхідними даними для аналізу в даній системі є кількість новинних згадувань того чи іншого ключового слова, за яким ведеться дослідження, у засобах масової інформації.

Такі дані зазвичай формуються при так званому моніторингу засобів масової інформації та інших джерел інформації таких, як соціальні мережі. Під кількістю згадувань мається на увазі кількість зустрічань ключового слова, що досліджується саме в цих новинних повідомленнях.

В такого виду часових рядах важливої інформацією є сезонність, врахування якої дозволяє також сильно підвищити точність.

Кількості зустрічань можна отримати багатьма способами, наприклад, використавши стандартні інтерфейси роботи з тими чи іншими аналітичними системами, що збирають інформації, вивантаживши їх у потрібному форматі або ж самому створити набір даних за допомогою таких способів, як парсинг даних або ж отримати доступ до API соціальних мереж, які в свою чергу не завжди можуть надати таку можливість. Приклад джерела інформації, який може бути використаний наведено нижче на рисунку 3.

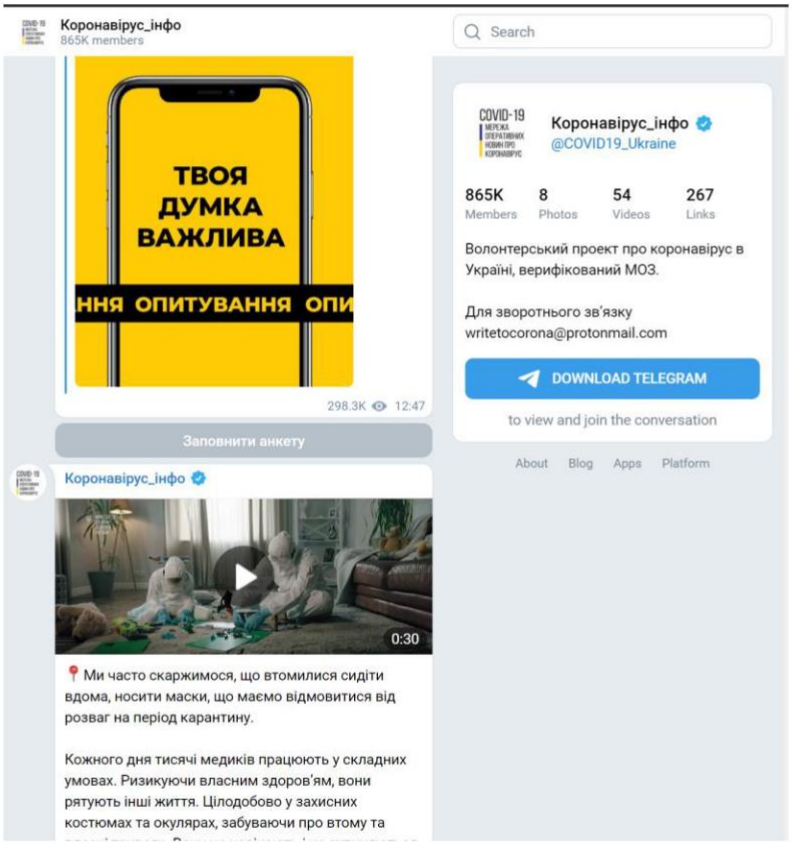

Рисунок 3 - Телеграм-канал «Коронавірус_інфо», який може бути використаний в якості джерела інформації

Наприклад, Телеграм не надає можливості отримання API без обмежень для доступу до своєї бази даних із повідомленнями, тому доводиться використовувати парсинг даних.

Парсер – програмний засіб для виділення інформації з вихідного html-коду веб-сторінки, який зображено на рисунку 4 за допомогою певних програмних засобів за певними правилами.

Парсинг даних – це отримання потрібної нам інформації з використанням парсерів. Періодичність оновлення, кількість, актуальність даних, що збираються парсером залежить від його програмної реалізації.

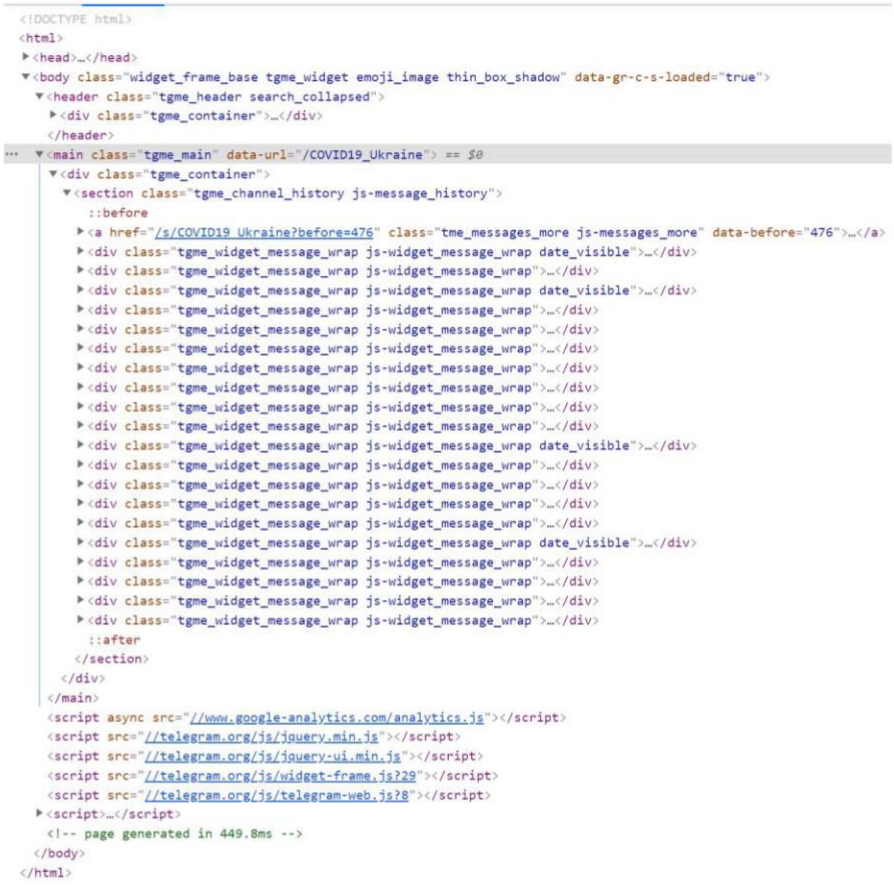

Рисунок 4 - Вихідний html-код веб-сторінки Телеграм-каналу «Коронавірус_інфо», який не має ніяких обмежень на збір даних з нього

Для отримання якісного сервісу із високим показником точності розпізнавання аномалій, дані повинні бути якісними та без викидів, спричинених якоюсь втратою даних, в таких аналітичних системах – це є неприпустимо, тому вхідні дані, їх збір і підготовка набору є дуже важливим етапом у такого виду аналітичних дослідженнях.[1]

Також особливістю даної системи є її миттєвість знаходження аномалій, адже подані дані аналізуються в реальному часі, що дає змогу використати цю систему в складі інших систем для, наприклад, оповіщення про появу аномалій в цьому ряді згадувань, так зване «алертування», що дає можливість користувачам системи швидко взнавати про якісь «вибухи» згадувань в засобах масової інформації чи соціальних мережах. [3]

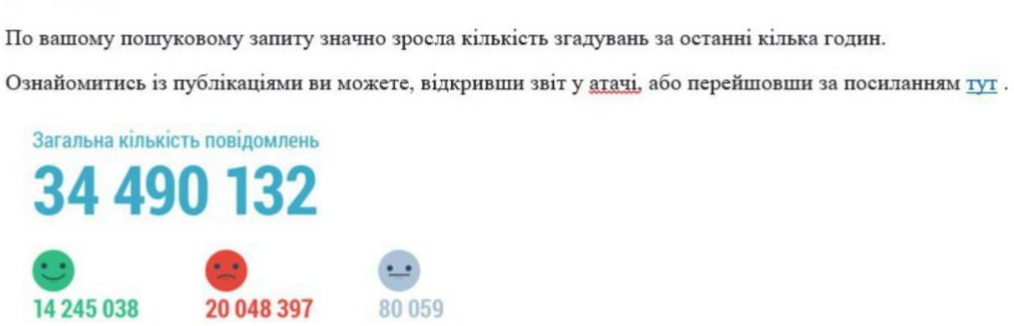

Рисунок 5 - Приклад так званого оповіщення у аналітичній системі, яка використовує дану систему виявлення аномалій

Такі оповіщення можуть відбуватись у вигляді електронного листа, як зображено на рисунку 5 із детальною інформацією про знайдену аномалію або у будь-який із месенджерів у вигляді повідомлення від бота.

1.2. Актуальність теми

Актуальність даної теми доволі висока у зв’язку із великою кількістю інформації, що генерується в засобах масової інформації та соціальних мережах, які неможливо відслідкувати і тим паче обробити, не озброївшись якоюсь аналітичною системою. А система знаходження аномалій серед часового ряду доволі сильно допомагає в цьому, що дає можливість миттєво знаходити будь-які інфоприводи, створені ключовим словом. Адже збільшення кількості зустрічань того чи іншого ключового слова в засобах масової інформації майже ніколи не проходить безпричинно.

Багато аналітичних систем вже використовують схожі системи виявлення аномалій. Серед них YouScan, LOOQME та SoMo, остання, до речі, використовує покращену версію розробленої системи.

1.3. Аналіз існуючих рішень

За останні роки було розроблено велику кількість вирішень даної задачі, проте більшість із них мають свої недоліки.

Виявлення аномалій в часових рядах не є якоюсь новою темою для вивчення, тому вже є кілька готових рішень, які варто розглянути. Ці рішення як знаходяться у публічному доступі, так і є приватними. Програмне забезпечення поділяється на відкрите та закрите.

Відкрите програмне забезпечення – це програмне забезпечення, вихідний код якого знаходить у відкритому доступі. Зазвичай код такого виду розповсюджується із так званими «вільними» ліцензіями, що дає змогу використувавати надбання цієї програми, а інколи і не лише надбання, для створення інших. Такого виду програмне забезпечення має в собі ряд переваг і недоліків. Зокрема, з переваг можна виділити підтримку спільнотою розробників, які постійно можуть допомогти у налагодженні коду та впровадженні нових функцій. Із основних недоліків можна відмітити неспроможність однієї людини якісно розробити систему на ранніх етапах до моменту, коли спільнота почне допомагати.

Закрите програмне забезпечення – це програмне забезпечння, код якого знаходиться у закритому доступі для інших розробників. Такий продукт розповсюджується лише під закритими ліцензіями, які не дозволяють вносити якісь правки або ж використовувати його задля розробки нових продуктів. Такий продукт із переваг має гарну підтримку на ранніх етапах, якість її роботи. З недоліків же можна назвати часту недобросовісність

власників продукту у вирішенні якихось невеличких поломок у зв'язку із зазвичай плановими оновленнями та роботами над продуктом.

Із основних аналогів, які представлені можна виділити наступні способи аналізу часових рядів з використанням аналітичних систем, що містять в собі такі функції або інших програмних засобів.

1.3.1. YouScan

Аналітична система, яка має в собі таку систему виявлення аномалій, проте як і більшість сучасних аналітичних систем використовує власні рішення, які, на жаль, знаходяться у закритому доступі.

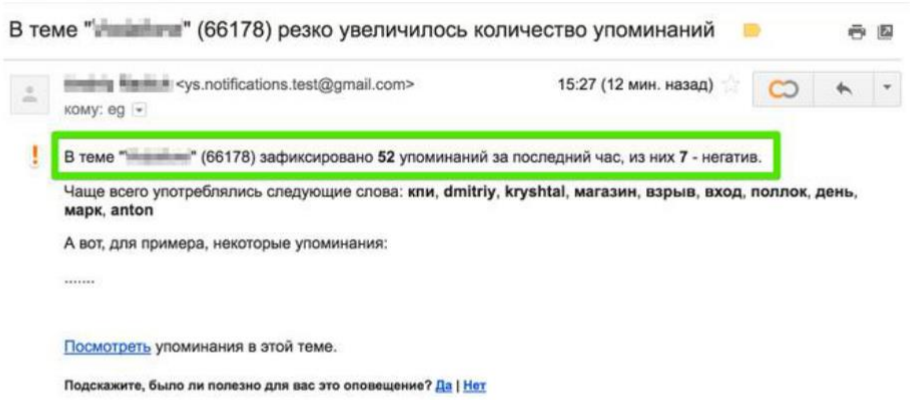

Рисунок 6 - Приклад оповіщення про результат виявлення аномалії за допомогою системи YouScan

Із тестового періоду використання системи лише можна відмітити коректність роботи своїх функцій, високу точність, простоту використання. Оскільки YouScan не дає доступу до використання цієї функції як окремої системи, більше інформації про роботу цієї підсистеми отримати неможливо. Із недоліків – відсутність можливості використати свої дані для аналізу та ціна користування такою системою, яку навряд може дозволити собі невелике аналітичне агенство.[6]

1.3.2. Microsoft Azure

Хмарний обчислювальний сервіс розроблений Microsoft, який дає змогу, не знаючи, мов програмування налаштовувати і створювати власні мініаналітичні системи, в тому числі з аналізом часових рядів. І навіть з використанням технологій машинного навчання, що дозволяє дуже сильно підвищити точність виявлення аномалій. На рисунку 7 наведеному нижче показано результат виявлення аномалій і форму, в якій Azure дозволяє їх отримати.

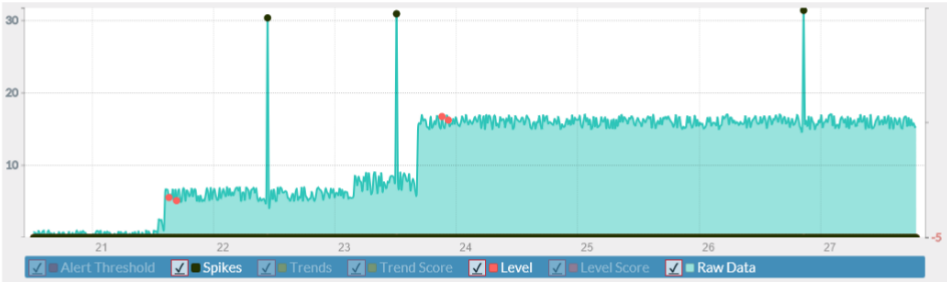

Рисунок 7 - Графік виявлення аномалій за допомогою Microsoft Azure

Із переваг – це зручний інтерфейс, швидкість, точність, широкий спектр функцій. Але із недоліків варто відзначити відсутність уже налаштованої системи, яка може бути використана у вирішенні нашої проблеми, а також плата за користування.[7]

1.3.3. Arauto

Web-орієнтована система для обробки лише часових рядів. Із основних переваг даної системи можна відмітити те, що вона є відритим програмним забезпеченням, що дає змогу використати її напрацювання та врахувати недоліки та дуже гарна документація у поєднанні із красиво та грамотно оформленим репозиторієм на ресурсі GitHub.

Рисунок 8 - Інтерфейс системи Arauto

Серед недоліків швидкодія, на що може впливати не дуже потужний сервер, на якому вона знаходиться та неорієнтованість на саме нашу задачу – аналіз часових рядів кількості новинних повідомлень, проте система дає змогу налаштувати себе під потрібні вам режими роботи та має величезну кількість налаштувань.[5]

1.3.4. Hctsa

Модуль для аналізу часових рядів із використанням технологій Matlab. Із переваг – висока швидкість аналізу даних, особливо, із використанням GPU та те, що модуль подається як відкрите програмне забезпечення, також можна використовувати на більшості популярних мовах програмуванн – Python, Java, C++. Із недоліків – неможливість використання як окрему аналітична система та відсутність інтерфейсів «спілкування» із модулем. [8]

1.4. Висновки по розділу

Потреба у недорогих, простих рішеннях такого виду аналізу даних у світі є досить великою, особливо можна помітити велику нестачу відкритого програмного забезпечення, що призводить до монополізації великими корпораціями цієї області використання комп'ютерного забезпечення.

Серед напрямків подальших досліджень було вибрано і зроблено акцент на математичну статистику, а саме класичні алгоритми пошуку аномалій в часових рядах, а не алгоритми машинного навчання, які є доволі навантаженими та ресурсозатратними.

2. ОБГРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТІВ ТА МЕТОДІВ РОЗРОБКИ

2.1. Алгоритм

У даному проекті було прийнято використовувати алгоритм рухомого середнього (moving average), але у деякій вдосконаленій формі.

Середнє рухоме – це алгоритм, який використовується для аналізу випадкових процесів, а також часових рядів.

Основна ідея алгоритму полягає у виділенні підмножини із всієї множини даних та аналізу її, а саме знаходження середнього серед цієї підмножини. [9]

Загальний випадок алгоритму середнього, що рухається, можна виразити за допомогою формули:

$$WWMA_t = \sum_{i=0}^{n-1} \omega_{t-i} * p_{t-i}, \quad (2)$$

де t та n – кількість парметрів функції, що обробляється, – вага параметру, p - значення функції віддалене на інтервалів від поточного.

Для базового алгоритму був узятий алгоритм простого середнього, що рухається (Simple Moving Average) – обчислення простого арифметичного середнього серед значень на певному відрізку:

$$SMA_t = \frac{1}{n}\sum_{i=0}^{n-1} p_{t-i}, \quad (3)$$

де t та n – кількість параметрів функції, що обробляється, – вага параметру, p – значення функції в точці $t - i$. Цей алгоритм призводить до згладжування функції, чим більший інтервал буде взятий, тим більше згладжування функції відбуватиметься.

Також було використані такі поняття із математичної статистики та теорії ймовірностей, як стандартне відхилення дискретної величини, математичне сподівання, правило трьох сигм. Оскільки часовий ряд є дискретною випадковою величиною.

Дискретна величина – це величина, яка набуває одного і лише одного значення, яке залежить від випадкових факторів та заздалегідь непередбачувана, при цьому кількість цих значень є злічувальною.

Математичне сподівання – це значення, яке є середньоочікуваним при великій кількості випробувань та обчислюється за формулою:

$$\mu = \sum p_j x_j \ , \qquad (4)$$

де p – це ймовірність події дискретної величини x_j.

Стандартне відхилення – це величина, яка показує розсіювання випадкової величини відносно математичного сподівання та виражається формулою:

$$\sigma = \sqrt{\sum p_i \, (x_i - \mu)^2} \, , \qquad (5)$$

де p – це ймовірність події дискретної величини x_i, μ – це математичне сподівання.

Після цього для пошуку аномалій в даних використовується правило трьох сигм, яке полягає в тому, що в межах однієї сигми в обидві сторони відносно центру розподілу лежить 68% всіх можливих значень нормального розподілу, в межах двох сигм – 95% та 98% - в межах трьох сигм. [4]

Експериментальним методом було досліджено, що всі значення, які лежать за межами двох сигм (2σ) є аномаліями майже у всіх випадках і їх прийнято вважати викидами.

Приклад вже згладженої функції вхідних даних, що зображені на рисунку 2, та діапазону σ та 2σ, виявлених аномалій можна побачити на рисунку 9 нижче.

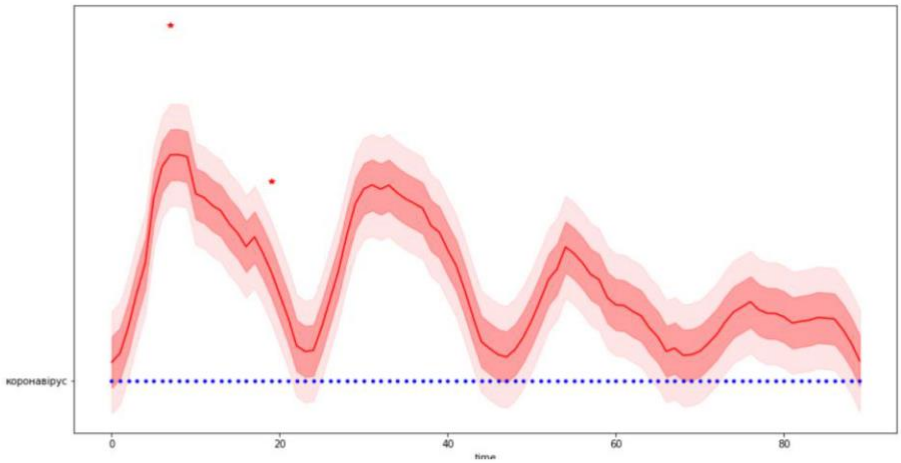

Рисунок 9 – Згладжений графік функції кількості згадувань слова «коронавірус» у ЗМІ за період з 16.04.2020 по 20.04.2020

Основною причиною використання такого алгоритму була потреба в простоті вирішення та не сильно вимогливій у ресурсах системі реалізовній у програмному засобі.[9]

2.2. Обгрунтування вибраної мови програмування для розробки

Невід'ємною частиною процесу розробки, яка вагомо впливає на кінцевий результат, є мова програмування. Вибір упав на скриптову інтерпретовану об'єктно-орієнотовану мову загального призначення Python.

Python – це вже доволі ненова мова, яка не вимагає особливого представлення її важливості в сучасному світі, адже її поширеність використання в науці про дані, яка зараз є однією із лідируючих, є дуже великою, навіть можна сказати, що Python монополізував цю область науки через свої переваги. [10]

До переваг Python можна віднести:

- простота встановлення;
- легкий та зрозумілий синтаксис;
- величезна підтримка з боку інших розробників;
- документація, яка автоматично генерується завдяки використанню певних конструкцій в коді

- швидка і легка розробка із застосуванням перевірених рішень.

До недоліків:

- повільність у порівнянні із компільованими мовами;
- так звані «Runtime Errors»;
- відсутність інтеграції із мобільними пристроями.

Через присутній досвід програмування та відлагоджування програм Python в поєднанні з його популярністю разом із відсутністю потреби запуску систему на мобільних пристроях була вибрана саме ця мова.

На рисунку 10 продемонстровано звичайну функцію, яка конвертує набір даних.

```
def get_counts_by_hours(user: dict, hours: int = 12, minutes: int = 60) -> pd.DataFrame:
    start = int(
        datetime.datetime.today().replace(
            minute=0, second=0, microsecond=0
        ).timestamp()) - 3600 * hours - 2 * 3600
    counts = pd.DataFrame(columns=['count', 'datetime'])

    for hour in range(0, hours):
        for _ in range(0, 60 // minutes):
            hour_count = get_count_filter(
                kw_body=[str(user['kw_body_code'])],
                jwt_token=user['jwt_token'],
                start=start, end=start + 60 * minutes
            )['count']
            hour_count = pd.DataFrame(
                data=[[hour_count, start]], columns=['count', 'datetime']
            )
            counts = counts.append(hour_count)
            start = start + minutes * 60
    counts = counts.reset_index(drop=True)
    return counts
```

Рисунок 10 – Приклад ділянки коду, який написаний на мові програмування Python

На рисунку 10 ми можемо побачити одну дуже важливу його властивість і особливість – це читабельність коду. Python – це не та мова, яка за десятки років (Python з'явився у далекому 1991 році) не дуже сильно еволюціонувала в напрямку швидкодії коду, але максимально, на мою думку, розвинулась і стала лідером у цій дуже важливій парадигмі нашої роботи

комп'ютерних інженерів. Перше, що має зрозуміти людина, яка починає створювати якусь систему, проект чи навіть невеличку програму на Python – це те, що «Код потрібно писати для людей, а не для машин».

Також на рисунку 11 можна помітити ще одну спробу допомогти в проблемі так званих «Runtime errors» – це анотації типів.

```
def get_counts_by_hours(user: dict, hours: int = 12, minutes: int = 60) -> pd.DataFrame:
    pass
```

Рисунок 11 – Приклад застосування анотації типів на функції написаній мовою Python

Помилки, що виникають під час виконання (Runtime errors) – це помилки, про які розробник не буде повідомлений інтерпретатором на етапі запуску програми, а лише під час виконання.

Анотації типів, які реалізовані в Python не змінюють виконання коду, але дають змогу розробникам зробити статичну перевірку типів за допомогою сторонніх програм та знову ж підвищують читабільність та легкість розуміння написаного коду.

2.3. Обгрунтування використаних фреймоворків і технологій

Основна частина системи була розроблена з використанням таких бібліотек мови Python та технологій:

- NumPy;
- SciPy;
- Pandas;
- Matplotlib;
- REST;
- FastAPI;
- Docker.

2.3.1. NumPy

NumPy – це бібліотека мови програмування Python, яка дає змогу обчислювати багатовимірні масиви в Python з застосуванням різних оптимізованих інструментів, які були використані при її розробці.[11]

Також важливо зауважити, що NumPy дає змогу виразити алгоритм, який часто в науці про дані записаний математичними формулами, операторами та операціями над даними, які знаходяться у вигляді матриці, за допомогою мови програмування Python при цьому не втративши у швидкодії в порівнянні з таким само алгоритмом, який відтворений тією ж «королевою швидкодії» мовою програмування C++ або C.

```
import numpy as np

x = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])
v = np.array([1, 0, 1])
vv = np.tile(v, (4, 1))
print(vv)
# Prints "[[1 0 1]
#          [1 0 1]
#          [1 0 1]
#          [1 0 1]]"
y = x + vv  # Add x and vv elementwise
print(y)
# Prints "[[ 2  2  4
#          [ 5  5  7]
#          [ 8  8 10]
#          [11 11 13]]"
```

Рисунок 12 – Приклад використання фреймворку NumPy

На рисунку 12 показана операція додавання матриць, яка реалізована в кількох рядках з врахуванням оголешення тих же матриць за допомогою NumPy і Python.

Із переваг відмітити варто: швидкість, універсальність, безальтернативність, простота, лаконічність.

За доволі довгий час роботи із цим модулем недоліків у нього так і не було виявлено, а швидкість реалізованого алгоритму навпаки була дуже високою на те вимогливою в ресурсах.

2.3.2. SciPy

SciPy – це фреймворк для виконання операцій чисельних методів та інших інженерних розрахунків над масивами NumPy. Разом із використанням цих масивів модуль SciPy також дуже сильно виділяється своєю швидкодією. Попри свою простоту використання, зустріти цей модуль нескладно в багатьох наукових та інженерних проектах, адже в ньому реалізовано дуже багато базових оптимізаційних алгоритмів і не лише. [14]

Рисунок 13 – Список доступних модулів у фреймоврку SciPy

На рисунку 13, користуючись командою «scipy?», можна отримати список модулів, які там знаходяться, зокрема нескладно помітити велику кількість функцій з різних видів математики – статистика, лінійна алгебра та інші.

Переваги: швидкодія, відносна простота в користуванні, читабельність, великий набір класичних математичних операцій.

Із недоліків можна відмітити невелике розмаїття в алгоритмах машинного навчання.

2.3.3. Pandas

Pandas – це пакет Python, написаний мовами програмування Python, Cython та мовою C, спроектований задля обробки даних, швидких маніпуляцій будь-якої складності, для зчитування те перетворення з великої кількості видів файлів, об’єктів, баз даних.[15]

Це фреймворк, заміни якому навряд знайдуть найближчі 5-10 років, адже Pandas дозволяє зчитувати і розпізнавати дані з таких видів об’єктів – файлів типу: «*.json», «*.xsl», «*.csv».

На рисунку 14 зображено розмаїття методів, які дозволяють брати дані практично з будь-яких популярних форматів зберігання даних.

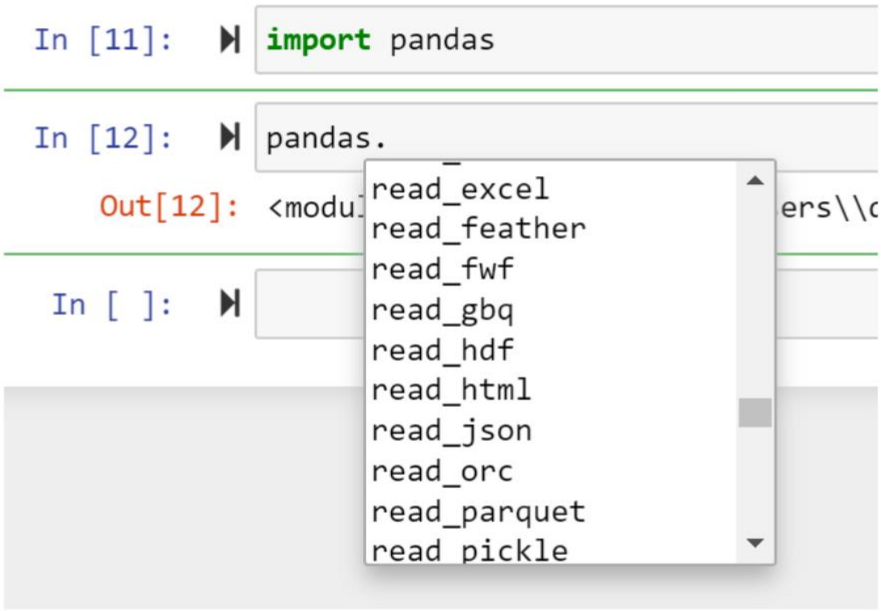

Рисунок 14 – Лише невеликий перелік модулів, що надаються пакетом обробки та трансформування даних Pandas

Зокрема варто виділити такі доступні методи:

- read_clipboard – зчитування з буфера обміну операційної системи, на якій запущено скрипт;
- read_csv – зчитування з файлу з даними, розділеними комою, «comma separated values»;
- read_html – зчитування даних з «сирого» html-коду;

- read_pickle – зчитування з серіалізованого об’єкту мови Python;
- read_sql_table – зчитування з SQL таблиці,

ці та інші методи є незамінними при роботі з даними.

Також в Pandas доступна функція не лише зчитування, а й запису в такі популярні види файлів, які на практиці використовують люди, які працюють в науці про дані – «*.csv», «*.xsl», «*.json», що дає змогу формувати доволі гнучкі аналітичні звіти із чітким та зрозумілим форматом.

Pandas має дуже високу швидкодію, це все завдяки схожій парадигмі, на які був створений модуль SciPy – це робота на основі масивів NumPy та їх ключової структури, яка має назву «DataFrame», що значно пришвидшує модуль та робить його нічим не гіршим за обробку даних на таких мовах програмування, як C/C++.

Переваги: величезний перелік форматів, з якими може працювати модуль, швидкість, гарно описана документація.

З недоліків можна відмітити не завжди зрозуміла навігація по осях та координатах таблиці, в якій представлені дані, проте цей недолік нівелюється завжди наочним та читабельним форматуванням при виводі на екран даних, що обробляються.

2.3.4. REST

REST (REpresentational State Transfer) – це стиль побудови архітектури програмних засобів, а саме комунікування один із одним за допомогою протоколу HTTP, головна відмінність від інших якого зводиться до того, що сервер не повинен запам’ятовувати стан клієнта між запитами.

Також REST архітектура зводить спілкування між сервісами до 4 основних типів операцій, зокрема:

- GET – запит, який відповідає за отримання даних із серверу, буквально в перекладі з англійської означає «отримати»;
- PUT – запит, який відповідає за модифікацію існуючих даних;

- POST – запит, який відповідає за створення нових даних на сервері;
- DELETE – запит, який відповідає за видалення даних з серверу, до якого звертається клієнт.

На рисунку 15 можна побачити приклад використання стилю побудови архітектури REST для налагодження повноцінної системи, яка складається з багатьох частин.

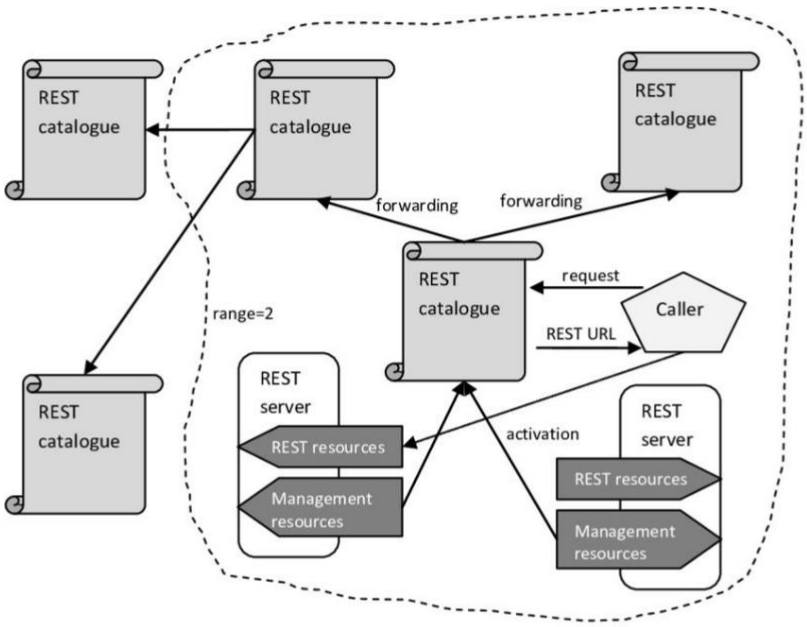

Рисунок 15 – Приклад використання REST сервісів для спілкування багатьох модулів системи між собою

Сервіс, який підтримує REST парадигму проектування програмних систем, називається RESTful.[13]

Також варто відмітити принцип шаровості, який полягає в тому, що сервіси певного шару можуть спілкуватись лише із сервісами цього шару.

2.3.5. FastAPI

FastAPI – це фреймворк, який створений для швидкої розробки високоефективних REST API.[13]

Пакет виділяється серед інших пакетів для створення подібних сервісів тим, що використовує останні напрацювання у швидкодії Python та його нові

можливості, зокрема, наприклад, асинхронність, яка підтримується в самому модулі (asyncio).

Також варто відмітити просто відмінну документацію із великою кількістю прикладів і способів використання в різних ситуаціях.

Окрім цих всіх плюсів FastAPI має в собі підтримку автоматичного генерування OpenAPI специфікації.

OpenAPI – це формат представлення API, завдяки якому можна візуалізувати RESTful API у наочно-зрозумілий для людини інтерфейс. Приклад візуалізованого API за стандартами OpenAPI зображений на рисунку 16 нижче.[13][20]

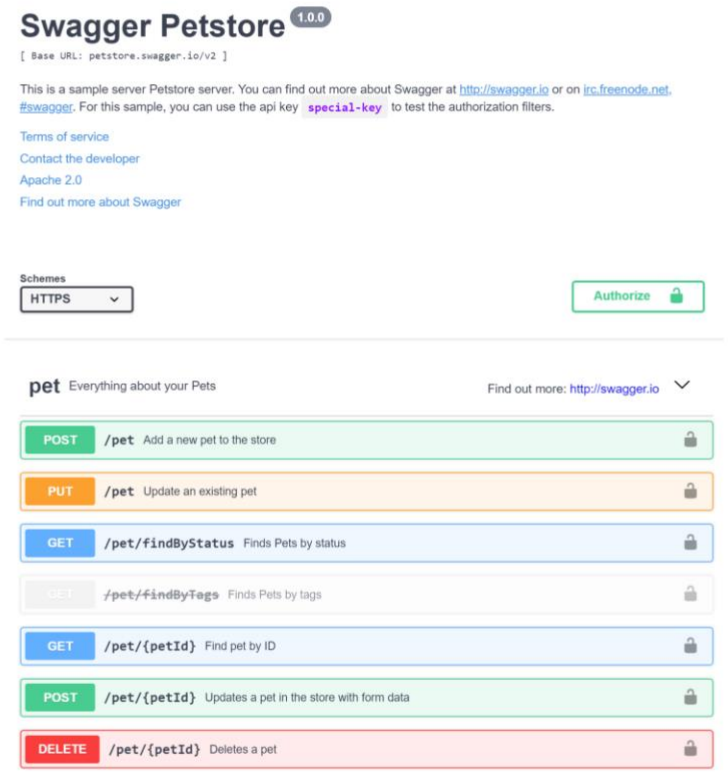

Рисунок 16 – Приклад візуалізації API за допомогою OpenAPI специфікації згенерованій у html-сторінку

На прикладі, показаному на рисунку 16 дуже гарно видно елегантність і водночас простота згенерованого інтерфейсу. Також із важливих функцій

для тестування системи іншими людьми в OpenAPI доступна функція тестування запитів прямо в інтерфейсі, так зване «Try it Out». На рисунку 17 можна побачити, що нам дається можливість потрібні нам поля у форматі json, завантажити файл, подивитися можливі варіанти відповіді від серверу, статуси помилок, формати об’єктів, що повертаються тими чи іншими методами.

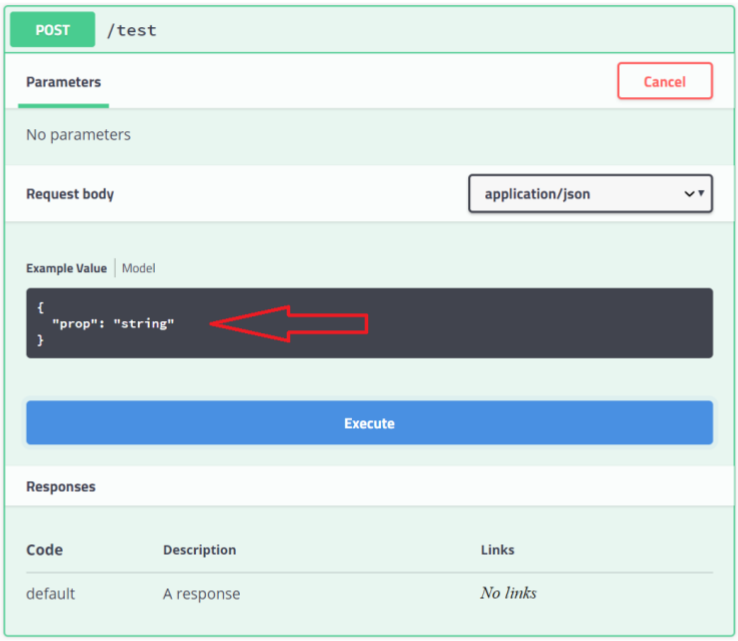

Рисунок 17 – Приклад тестування запитів до API за допомогою функції «Try it out», червона стрілка вказує на параметр, який може бути вказаний при запиті

Тому, використовуючи OpenAPI, на виході можна отримати вже повноцінний сервіс із підготовленою документацією, який можна використовувати надалі навіть як інтерфейс для користування системою.

Із недоліків варто відмітити вимогливість у описуванні всіх типів та моделей даних, які використовуються.

2.3.6. Docker

Для розгортання на зовнішніх сервісах було прийнято використовувати Docker.

Docker – це система розгортання програм та їх управління за допомогою контейнеризації, що означає, що вона використовує такі структури, як образи.[12]

Образ(image) – це стандартизований пакет Docker, який містить все необхідне для розгортання та запуску програми – бібліотеки, системні програми, засоби для роботи з мережою, якщо вони потрібні та інші залежності, запуск без яких не зможе відбутись. Всі налаштування та, що буде містити в результаті образ прописуються в окремому конфігураційному файлі Dockerfile. Далі можна мати можливість запустити цей образ та створити контейнер, що є, в загальному, вже запущений образ проекту.

Тобто Docker дає можливість публікувати нашу систему, як модуль, який можна підключити в іншу систему, принцип сервісів підтримується, отже може бути застосована мікросервісна архітектура. В загальному Docker також дає нам можливість підключати нашу систему в якості модуля у велику кількість сучасних систем, які розгортаються на таких сервісах, як Kubernetis, AWS та інших хмарних середовищах.[12]

Також Docker має в собі невід’ємну його властивість – це принцип шаровості, що дає змогу вести історію розробки проекту, його стадії, що нагадує роботу з вже підготовленим програмним засобом запакованим в образ(image) так само зручно, як з репозиторієм Github.

Також варто зауважити, що Docker і віртуальна машина – це абсолютно різні речі. Різницю можна побачити на схемі на рисунку 18.

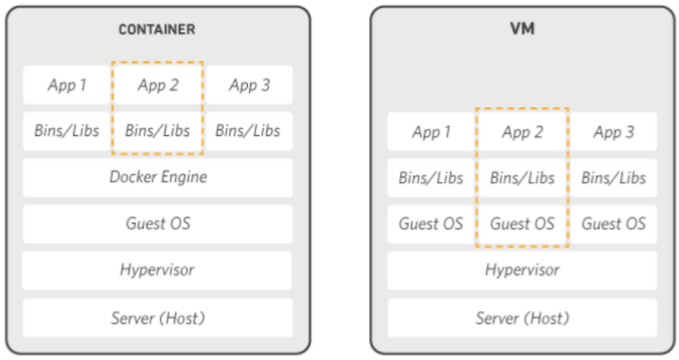

Рисунок 18 – Спрощена схема відмінностей віртуальної машини від контейнера Docker

Важливою особливістю Docker є можливість поставити всередину образу будь-яку операційну систему або її модифікацію і запускати всередині програмний продукт.

На рисунку 19 показана основна перевага такого способу поширення програмних засобів за допомогою Docker – це абсолютно безконфліткне існування кількості програм, які обмеженні лише можливостями самого Docker.

Рисунок 19 – Приклад основних команд використання Docker в командному рядку операційної системи Linux

Отже з переваг – це універсальність та розширення можливостей інтегрування та використання програмного засобу в інших системах, із

недоліків – відсутність так званого оркестратора керування навантаженням та розподіленням.

2.4. Система контролю версій

Для забезпечення збереженості лістингу коду проекту, що розробляється, в поєднанні з потребою в отриманні чіткої хронології розробки системи прийнято рішення використати систему контролю версій Git в поєднанні з віддаленим хостингом її за допомогою сервісу GitHub.

Git – це система котролю версій, яка розроблена Лінусом Торвальдсом, людиною, котра створила продукт, якому нема альтернатив – Linux, тому вибирати не було потреби, що ж із віддаленим сайтом-хостингом, на якому велася б віддалена система версій контролю, то вибір впав на GitHub, який зараз знаходиться у власності Microsoft, у зв’язку зі стабільністю та надійністю за роки використання у особистій та комерційній розробці проектів.[16]

2.5. Інтегроване середовище розробки PyCharm

Оскільки середовище розробки програмного забезпечення вибирається на власний розсуд програміста згідно його потреб та нюансів, на які він звертає увагу, було вибрано середовище PyCharm, яке дозволяє вести професійну розробку та підтримує велику кількість розширень.[17]

Була вибрана версія Professional Edition з використанням студентського акаунту університету у зв’язку із тим, що професійна версія містить в собі багато інструментів, які полегшують розробку, зокрема інтелектуальний алгоритм «підсвітки» коду, що прискорює написання лістингу коду, зменшуючи при цьому ймовірність помилок, також він має вбудований статичну програму для перевірки всіх типів на валідність, що дає змогу зменшити ймовірність «Runtime Errors».

На рисунку 20 можна побачити зручний інтерфейс середовища із зрозумілим та чітким виділенням різними кольорами різних частин коду.

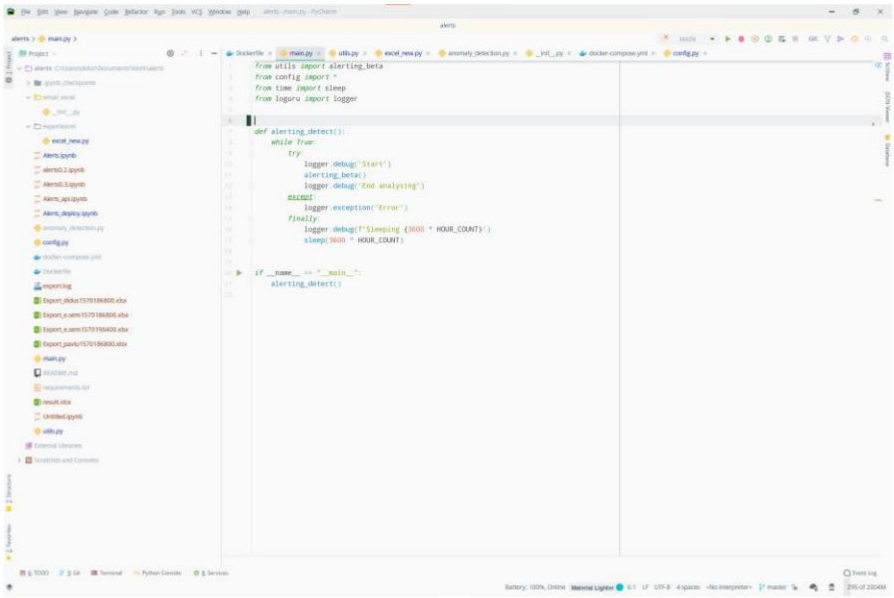

Рисунок 20 – Середовище розробки PyCharm

Із переваг варто відзначити:

- вбудований термінал для роботи з командним рядком;
- будь-яка популярна система контролю версій коду (Git);
- високотехнологічний відлагоджувач;
- автозбереження та забезпечення цілісності незбереженого коду;
- ергономічний інтерфейс;
- величезна кількість інтуїтивно зрозумілих у використанні функцій для роботи з проектом;
- робота з Docker’ом.

Із недоліків:

- присутність платної професійної версії, але вона доступна студентам;
- доволі високі вимоги до параметрів комп’ютера, на якому запускається.

2.6. Обгрунтування вибору операційної системи

Оскільки вибір операційної системи залежить від багатьох факторів, таких, як масовість, потреби користувачів системи, то вибір впав на розробку для Linux в Docker контейнері, що дозволило б стерти якісь обмеження на запуск системи, адже Docker можна запустити на всіх операційних системах.

Під час розробки використовувалось поєднання Windows та її технології WSL (Windows Subsystem for Linux), яка є шаром для запуску більшості програм Linux на Windows. Функція WSL дає змогу розробнику не відчувати ніякого дискомфорту, використовуючи Windows для розробки під ОС Linux. На рисунку 21 зображено приклад простоти використання командного рядка ОС Linux під керуванням Windows.

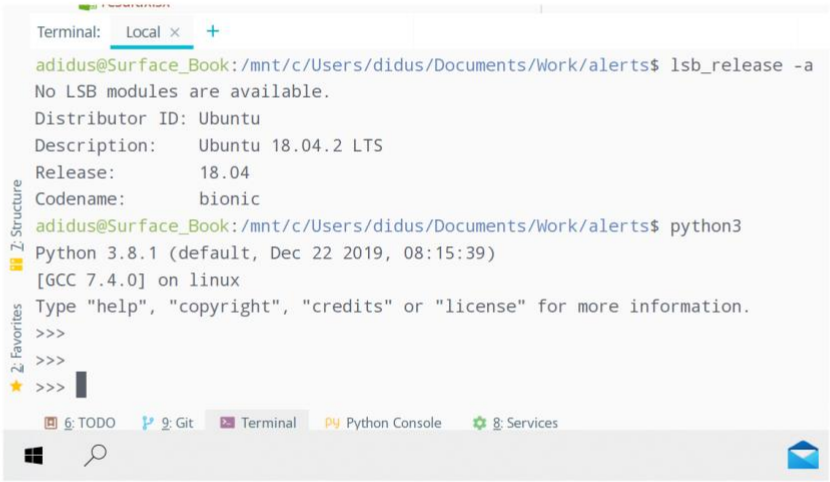

Рисунок 21 – приклад використання WSL

2.7. Висновки до розділу

Було прийнято рішення використовувати власно-модифікований алгоритм рухомого середнього, що дає змогу не витрачати велику кількість ресурсів та знаходити швидко аномалії в часових рядах зустрічань новинних повідомлень.

З обгрунтованих причин було прийнято рішення використовувати мову програмування Python в поєднанні з ОС Linux, а задля універсальності було

вибрано метод розгортання аналітичної системи за допомогою системи віртуалізації Docker. Ведення

Також було використано класичний стек для розробки систем для статистичної обробки мовою Python– NumPy, Pandas, SciPy, matplotlib.

Всі ці параметри здатні забезпечити швидку, сучасну та зручну розробку програмного реалізації аналітичної системи.

3. АРХІТЕКТУРНІ ТА СТРУКТУРНІ РІШЕННЯ, ОПИС РЕАЛІЗАЦІЇ СИСТЕМИ В ПРОГРАМНОМУ ЗАСТОСУНКУ

3.1. Архітектура системи

В сучасному світі протягом останніх років формується тенденція до використання мікросервісної розподіленої архітектури і цьому є дуже багато причин.

Варто розпочати з того, що мікросервісна архітектура як термін, як визначення, як об’єкт вивчення комп’ютерних наук з’явився відносно недавно – лише в 10-их роках 21-ого століття, цьому передувало дуже багато причин, зокрема поширення веб-додатків, попит на яких ріс з кожним роком все більше, і старі веб-додатки повинні мали мати можливість конкурувати з конкурентами, які з’являлись щодня.

Тому серед важливих критерії програмного забезпечення з’явилась швидкість розробки та простота внесення змін в роботу додатку, тобто гнучкість, що спричинило появу мікросервісної архітектури у супротив до вже усталої на той час монолітної архітектури.

Монолітна архітектура – це спосіб розробки програмного забезпечення, при якому всі компоненти системи тісно взаємодіють між собою та є зазвичай невід’ємними та недієздатними при відокремленні.

Мікросервісна архітектура – це спосіб розробки програмного додатку, системи чи засобу з використанням парадигми, що програмне забезпечення складається з набору невеликих незалежних програмних модулів (їх називають сервіси), які можуть працювати окремо або за відсутності одного і більше сервісів не припиняють свою роботу. Також для спілкування між собою сервіси використовують такі механізми передачі даних, як HTTP, AMQP та інші аналогічні мережеві протоколи.[19]

Серед позитивних сторін використання такої архітектури є:

- завадостійкість;

- краща поведінка під час пікових навантажень, ніж у монолітної побудови;
- швидкість розробки;
- сучасність технологій, які можна застосувати;
- розгортання і гарна сумісність із хмарними обчислювальними сервісами
- легше впровадження та підключення нових сервісів у систему.

Із недоліків варто відмітити:

- високі вимоги до мережі;
- складність підтримки (потрібні DevOps);
- складність у тестуванні, особливо інтеграційних тестах.

На рисунку 22 можна побачити схематичне зображення такого виду архітектур

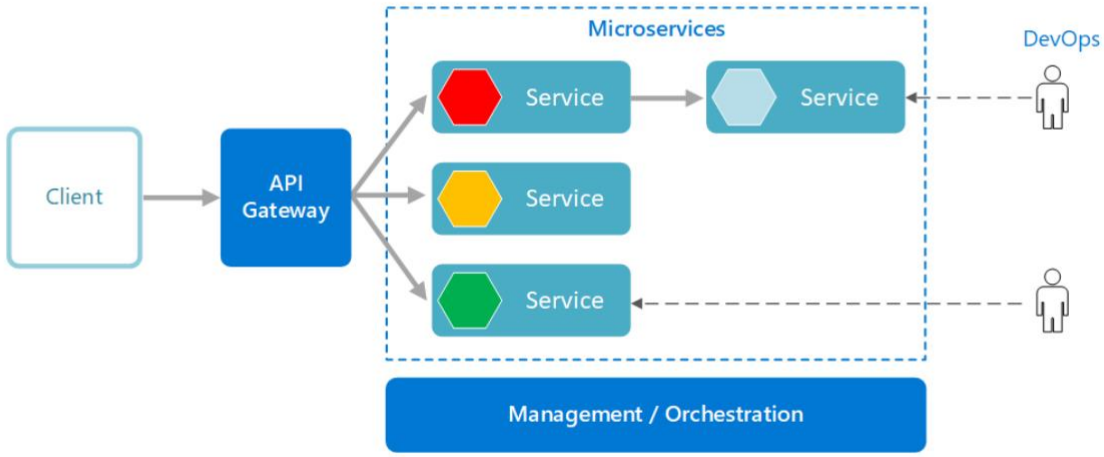

Рисунок 22 – Приклад мікросервісної архітектури

Тому для розробки було вибрано розробити систему у вигляді сервісу, який зроблений по всіх канонах мікросервісної архітектури.

3.2. Структура реалізації алгоритму системи в програмному засобу

Для реалізації аналітичного алгоритму було прийнято створити проект з основними ключовими частинами для роботи повноцінного сервісу на основі Docker.

Структура файлів та директорій проекту вийшла наступна (зображена на рисунку 23):

```
anomaly_detector/
├── Dockerfile
├── app
│   ├── anomaly_detector
│   │   ├── anomaly_detector.py
│   │   └── utils.py
│   ├── config.py
│   └── main.py
├── docker-compose.yml
└── requirements.txt
```

Рисунок 23 – Структура програмного засобу

Даний сервіс включає в себе 2 директорії та 7 файлів.

3.2.1. Контейнеризаційний файл Docker

Конфігураційний файл Docker потрібен в даному проекті для створення образу контейнерного среедовища Docker. Він складається з набору команд, які зображені на рисунку 24:

```
1  FROM python:3.8
2
3  EXPOSE 80
4
5  WORKDIR /anomaly_detector
6  COPY ./requirements.txt .
7
8  RUN pip install -r requirements.txt
9
10 COPY . /anomaly_detector
11
12
```

Рисунок 24 – Структура файлу Docker

Даний образ буде наслідуватись із образу python:3.8, що можна побачити на рисунку 24 в рядку 1, оскільки при розробці та внесенні правок у виконувані файли системи потрібно буде заново перезбирати образ, то команди потрібно розставляти саме у даній послідовності, адже Docker підтримує принцип «шаровості» та кешованості, що пришвидшує його роботу при повторній будь-якій збірці.

3.2.2. Модуль app

Модуль app складається з модуля з назвою anomaly_detector, в якому знаходиться клас реалізації алгоритму пошуку аномалій.

Складається з двох файлів:

- anomaly_detector.py;
- utils.py.

Файл з назвою anomaly_detector.py відповідає за реалізацію самого алгоритму, його основні робочі моменти, а файл з усталеною назвою utils.py відповідає за різні службові функції, які було прийнято винести в окремий файл для кращої читабельності та розвантаженості коду.

У файлі config.py знаходяться глобільні конфігураційні змінні, які зчитуються зі змінних середовища, які прописуються відповідно в docker-compose. У config.py можна помітити, що змінні зберігаються у закритому вигляді, це зроблено для можливості публікування конфігураційних файлів у віддалених репозиторіях.

Файл main.py відповідає за підняття свого API, в ньому знаходяться прописані шляхи методів та налаштування FastAPI. Цей файл являється вхідною точкою системи, тому він є теж дуже важливим.

3.2.3. Композиційний файл інструменту Docker Compose

Docker Compose – це інструмент для розгортання багатомодульних систем із зручною мовою конфігураційного файлу з великим інструментальним розмаїттям, зокрема на рисунку 25 можна побачити такі використані інструменти, як:

- «прокидання» портів (port-forwarding) (7-8 рядок);
- змінні середовища (environment variables) (9-15 рядок);
- команда запуску контейнера (start command) (16 рядок).

```
1   version: '2'
2
3   services:
4
5     alerter:
6       build: .
7       ports:
8         - "7576:5000"
9       environment:
10        - DOMAIN=http://www.example.com
11        - EMAIL_USERNAME=example@example.com
12        - EMAIL_PASSWORD=password
13        - PROBABILITY=0.7
14        - URL=url.com
15        - HOUR_COUNT=3
16      command: uvicorn app.start:app --host=0.0.0.0 --port=5000
17
```

Рисунок 25 – Конфігураційний файл docker-compose.yml

Завдяки використанню Docker Compose, ми максимальнос спрощуємо будь-які маніпуляції з контейнером та його налаштуванням на віддаленому сервері.[18]

3.2.4. Файл залежностей requirements.txt

Файл залежностей використовується задля налаштування локального інтерпретатора Python, що лежить в контейнері. За допомогою виконання команди в командному рядку *pip install -r requirements.txt* можна отримати всі потрібні бібліотеки (зображені на рисунку 26) для запуску даної системи.

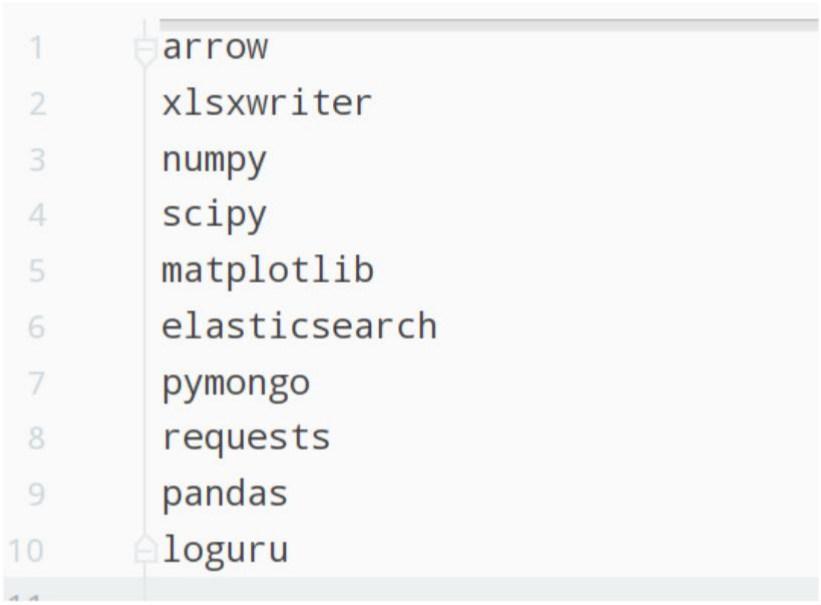

Рисунок 26 – Файл залежностей requirements.txt

3.3. Методи розгортання системи

Для розгортання даної системи можна використати два способи:

- за допомогою технології Docker;
- з використанням локального інтерпретатора Python

3.3.1. Використання контейнеризаційної системи

Для запуску системи у якості самостійного сервісу або складової якоїсь аналітичної системи потрібно:

1) склонувати репозиторій на машину, де збирається проводитись розгортання;
2) відредактувати файл docker-compose.yml, виставивши власні змінні середовища та порти;
3) прописати команду docker-compose up -d, яка завантажить та збере всі необхідні образи для роботи системи у вигляді демону, який залишатиметься завжди запущеним;
4) для зупинки сервісу використовується команда docker-compose down, яка зупиняє всі контейнери та видляє їх, залишаючи зібрані образи;
5) при внесенні змін до виконуваних файлів, потрібно виконувати команду docker-compose up -d –build, яка виконуватиме повне перезбирання сервісу з врахуванням зроблених змін.

Такий спосіб розгортання є повністю коректним та більш правильним за другий, особливо, при використанні сервісу, як складова іншої комерційної аналітичної системи.

3.3.2. Використання локального інтерпретатора

Цей спосіб дуже рекомендується використовувати, як тимчасове, локальне вирішення за неможливості встановлення Docker або для використання у якості відлагоджування.

Для такого варіанту також краще використовувати віртуальне середовище Python (venv), завдяки якому відбувається лише локальна інсталяція в саме той інтерпертатор, без нагромадження зайвих бібліотек у системному інтерпретаторі.

Для розгортання на інтерпретаторі Python потрібно виконати такі кроки в комнадному рядку в папці з проектом:

1) виконати команду pip install -r requirements.txt;
2) прописати у ваші змінні середовища системи змінні, які вказані в файлі docker-compose.yml;
3) виконати команду uvicorn app.main:app --host=0.0.0.0 --port=5000 з портом, який потрібно

3.4. Способи використання

До системи ставились вимоги бути доступною для користування, швидкою та універсальною.

Системою можна користуватись двома способами:

- з використанням посторонніх бібліотек мов програмування для виконання запитів;
- за допомогою браузера.

3.4.1. Використання бібліотек для виконання запитів

Для доступу до системи в якості API можна використовувати бібліотеки для виконання запитів.

Для прикладу можна використовувати такі мови програмування і бібліотеки для доступу до API:

- Python – requests, http.client;
- Java – OkHttp, Unirest;
- JavaScript – Fetch, jQuery;
- Go – Native;
- C# - RestSharp;
- C – libcurl;
- та ін.

На рисунку 27 зображено приклад ділянки коду, за допомогою якої можна отримати доступ до API системи з використанням мови програмування Python та бібліотеки requests.

```python
import requests

url = "http://localhost:8000/anomaly_detect"

payload = "{\"count\":{},\"timestamp\":{},\"kw_body\":{}}"
headers = {
  'accept': 'application/json',
  'Content-Type': 'application/json',
  'Content-Type': 'text/plain'
}

response = requests.request("POST", url, headers=headers, data_=_payload)
```

Рисунок 27 – Приклад доступу до системи з використанням мови програмування Python

Також у зв'язку з неймовірною популярністю мови програмування JavaScript на рисунку 28 зображено приклад доступу цією мовою до системи.

```javascript
var settings = {
  "url": "http://localhost:8000/anomaly_detect",
  "method": "POST",
  "timeout": 0,
  "headers": {
    "accept": "application/json",
    "Content-Type": ["application/json", "text/plain"]
  },
  "data": "{\"count\":{},\"timestamp\":{},\"kw_body\":{}}",
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

Рисунок 28 – Приклад доступу до системи мовою програмування JavaScript з використанням jQuery

3.4.2. Використання браузера

Іншим варіантом доступу є доступ за допомогою браузера та інтерфейсної частини системи. Цей варіант є більш «цивільним» та звичайним для простої людини.

Для використання системи можна використовувати автоматично згенерований інтерфейс бібліотеки FastAPI за канонами OpenAPI, який дозволяє використовувати її як окремий повноцінний web-додаток, лише надавши системі необхідні та валідні вхідні дані. Приклад згенерованого інтерфейсу показани на рисунку 29 нижче.

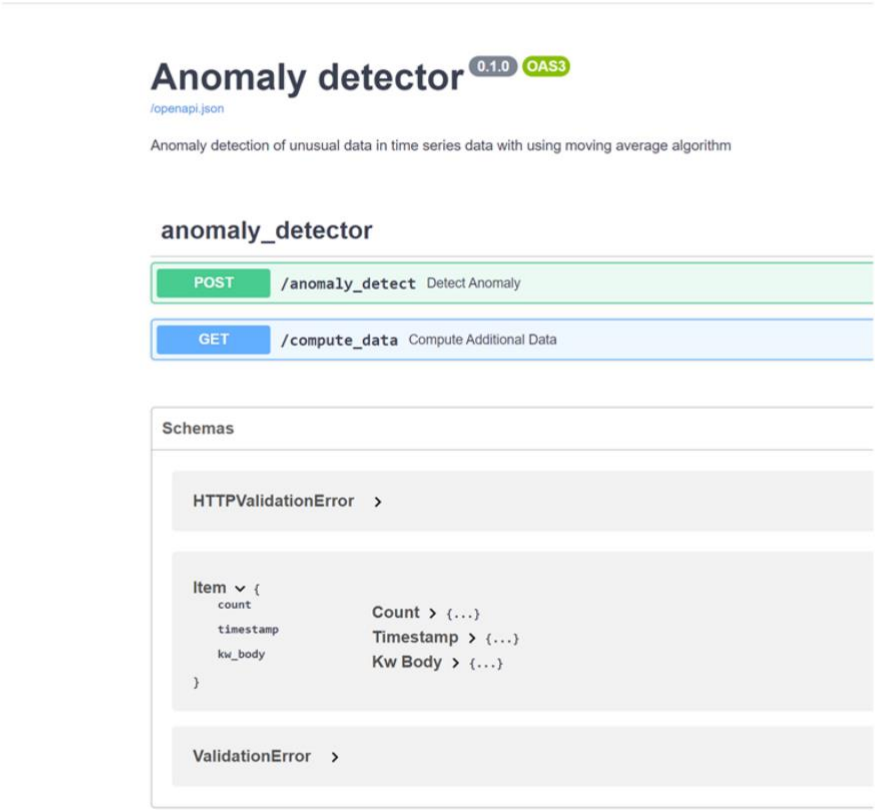

Рисунок 29 – Інтерфейс системи для користування

Такий інтерфейс можна виокористовувати в якості інтерфейсу:

- тестування системи;
- ознайомлення з системою та її параметрами;

- автоматичної документації для розробників.

Такий інтерфейс є доволі зрозумілим та чітким, оскільки чітко вказує на формат вхідних даних та вихідних, кои помилок та іншу корисну інформацію. Зокрема, на рисунку 30 для прикладу зображена схема даних, які використовуються в якості вхідних.

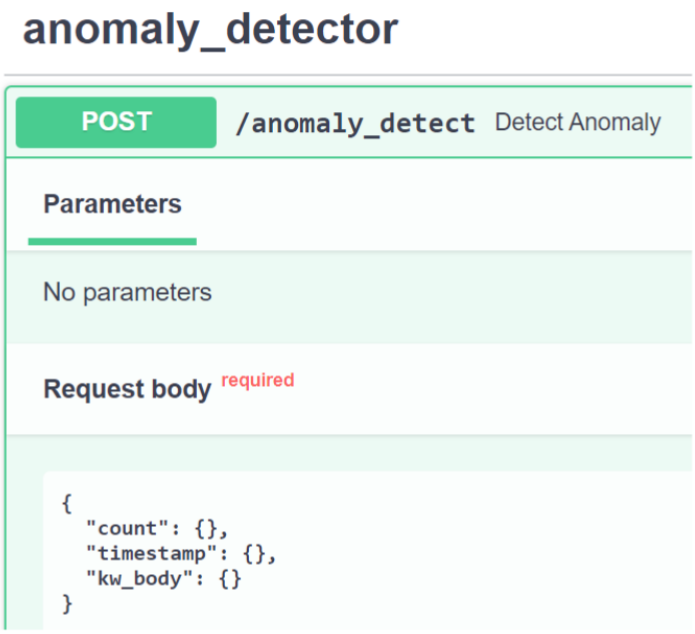

Рисунок 30 – Приклад того, як вказується формат схеми вхідних даних

3.5. Висновки до розділу

Вибрані інструменти дозволили побудувати сучасну, оптимізовану та стійку до навантажень архітектуру. Серед розглянутих та досліджених варіантів методика мікросервісної архітектури зарекомендувала себе з гарного боку маючи багато сильних сторін.

Способи розгортання системи, зокрема контейнеризаційний спосіб, дозволяють швидко, зручно та майже будь-де розгорнути зліпок системи в тому вигляді, в якому він був створений.

А інтерфейс доступу до системи, який дає змогу не лише отримувати інформацію з браузера, а й імплементувати її в якості підсистеми в інші аналітичні системи.

4. ТЕСТУВАННЯ ТА РОЗРОБЛЕНІ СПОСОБИ ВДОСКОНАЛЕННЯ ДАНОЇ СИСТЕМИ

4.1. Тестування системи

Так як алгоритм важко протестувати, а лише можна спостерігати за його роботою та з часом виявляти його слабкі сторони, тому було вирішено протестувати на час виявлення аномалій серед стандартизованих даних без явних якихось відмінностей від інших наборів даних.

Було використано вбудовану «магічну» команду «%%time» в Jupyter Notebook – це одне із середовищ розробки мовою Python, приклад тестування та виданого часу показано на рисунку 31 нижче.

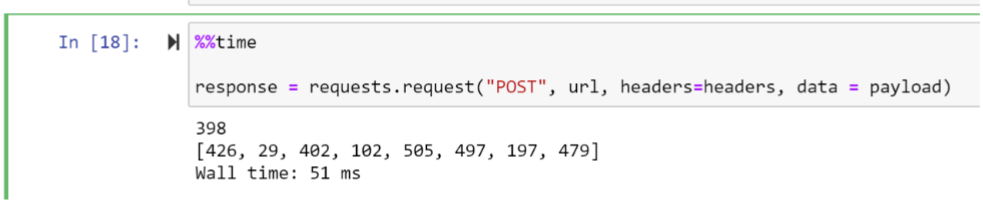

Рисунок 31 – Приклад тестування

Результати тестування на час виконання в залежності від розміру масиву наведені на таблиці 1 нижче.

Таблиця 1 – Результати тестування швидкодії системи

Ключове слово	Розмір даних (кількість елементів)	Середній час
«коронавірус»	90	51 ms
«коронавірус»	262	52 ms
«коронавірус»	398	51 ms
«україна»	89	54 ms

«україна»	194	55 ms

Також для тестування було виконано навантужавальне тестування, в результаті якого перевірялась система на стійкість до інтенсивних і високочастотних запитів.

Для виконання такого тесту було написано скрипт для запуску постійних паралельних запитів до API системи, після запуску якого система вистояла та зберегла свою роботоздатність.

Варто зауважити, що збільшився час відповіді на запит для пошуку аномалій, зокрема, при постійних паралельних запитах результати тестування на час стали такими, які наведені у таблиці 2.

Таблиця 2 - Результати тестування швидкодії системи при постійній завантаженості системи іншими запитами

Ключове слово	Розмір даних (кількість елементів)	Середній час
«коронавірус»	90	101 ms
«україна»	194	125 ms

Також для тестування в ході розробки обов’язково застосовувалось так зване димне тестування, яке відповідає за саму роботоздатність системи та відсутність помилок на етапі розгортання.

4.2. Способи майбутнього вдосконалення

До важливих якостей системи належить також і можливість її вдосконалення, зокрема повторне використання коду, можливі механізми заміни модулів та інші дії, які можна виконувати в ході оновлення.

Ця система розроблена з відразу спланованою структурою та ядром, яким є алгоритм, навколо якого все і працює. Є кілька варіантів розвитку оновлень та вдосконалень системи:

- покращення інтерфейсу;
- зміна алгоритму;
- застосування машинного навчання та нейронних мереж;
- підвищення стійкості до великих навантажень;
- розширення кросплатформенності.

Важливою стороною поліпшення є інтерфейс, адже зрозумілість, ергономічність, інформативність системи є її важливим аспектом. Відтак для поліпшення системи було б добре розробити та написати окремий інтерфейс із власним стилем. Проте це буде потребувати ресурсних затрат, що може заповільнити її.

Щодо застосування машинного навчання, то воно має дві сторони: перша сторона – це покращення розпізнавання, пришвидшення, а з іншої – це більша витрата ресурсів, як при розробці так і при використанні вже готової системи, різниця лише в тмоу. Що там буде витрата в людських ресурсах, а там в процесорному часі та пам’яті.

Щодо стійкості до навантажень, то одним із варіантів може бути розпаралелювання та додавання асинхронності, де це можливо, проте це буде дієвим лише з використанням багатоядерних процесорів, також складністю з розпаралелюванням стане GIL – Global Interpreter Lock в мові програмування Python, це механізм, який в один момент, квант часу дає змогу працювати лише одному процесу. Для уникнення цього можна використати різні рішення, проте доцільним в такому випадку буде зміна мови на такі, які підтримують, наприклад Golang.[21]

Щодо кросплатформенності, то із непідтримуючих пристроїв, на яких не можна розгортати дану систему, залишились лише мобільні пристрої, смартфони, проте використання даної системи доступне, адже інтерфейс

спілкування доступний з будь-якого браузера, проте виникає питання в ергономічності та зручності користування, показники яких точно поліпшив би окремий мобільний додаток під різні операційні системи – як Android, так iOS

4.3. Висновки до розділу

Тестування даної системи показало, що швидкодія розробленого сервісу є доволі високою та не залижить від розмірів вхідних даних, проте сильно залежить від кількості клієнтів, які в один момент під’єднуються.

Високі показники швидкодії підтвердили коректне використання даного алгоритму.

Щодо способів поліпшення цієї системи, то одним із вагомих таких кроків заради вдосконалення було б перенесення її на мобільні пристрої.

ВИСНОВКИ

Серед головних проблем аналітичних систем, які спеціалізуються на пошуку аномалій в часових рядах є:

- закрите програмне забезпечення;
- використання ресурснозатратних алгоритмів.

Ці проблеми і були розглянуті у першому розділі дипломного проекту на прикладі аналогів та конкурентів, які надають такі послуги. Часто проблема ресурснозатратності постає перед невеликими стартапами, які потребують простих, швидких та стійких до навантажень систем, які можна вбудувати в якосты модуля у власну розроблювану аналітичну систему або інший продукт, що розробляють.

Перед проектуванням та реалізацією системи було пророблено дослідження та переглянуто основні негативні сторони в уже працюючих аналогах. Такого виду дослідження є звичним для проектування навантажених систем, тому в ході його було зроблено висновки, що використання алгоритмів машинного навчання буде необгрунтованим, адже однією із вимог побудови системи була швидкодія. Проте важливими перевагами аналогів системи були стабільність, інтерфейс роботи з ними, модулі оповіщення користувача про аномалії та інші користувацькі корисні функції.

При побудові архітектури системи важливу роль відігравала простота розгортання та поширення системи, які забезпечила контейнеризаційна система Docker. Вже в реалізованій системі це не раз забезпечувало максимальну швидкість розгортання нової такої ж системи на інших сервера та гнучкість роботи з нею.

Інтерфейс доступу до системи вийшов доволі аскетичним, проте інформативним, також доступ до системи був запезпечений за допомогою API, при димному тестуванні система постійно забезпечувала валідні відповіді знайдених аномалій.

Також в ході тестування було підтверджено відсутність змін у часі аналізу новинних повідомлень у відповідь на збільшення розмірів масивів вхідних даних, що було доволі важливим, також модуль FastAPI показав себе, як гарно збалансований з високим порогом відмови базовий рівень системи, що витримує навіть велику кількість запитів паралельно.

В ході виконання дипломного проекту було здобуто не лише систему виявлення аномалій, а й навички проектування систем такого виду з подальшим розробленням плану вдосконалень її.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Time Series [Електронний ресурс]. – Режим доступу до ресурсу: https://www.investopedia.com/terms/t/timeseries.asp
2. Anomaly detection [Електронний ресурс]. - Режим доступу до ресурсу: https://towardsdatascience.com/anomaly-detection-with-time-series-forecasting-c34c6d04b24a
3. Anomaly Detection system [Електронний ресурс]. - Режим доступу до ресурсу: https://medium.com/pinterest-engineering/building-a-real-time-anomaly-detection-system-for-time-series-at-pinterest-a833e6856ddd
4. Сапсай Т.Г., Сущук-Слюсаренко В.І. Основи теорії ймовірностей.-К.: НТУУ «КПІ», 2014. – 204с
5. Arauto [Електронний ресурс]. – Режим доступу до ресурсу: https://towardsdatascience.com/introducing-arauto-an-interactive-tool-for-time-series-forecasting-14b3e36b5f81
6. YouScan Alerts [Електронний ресурс]. – Режим доступу до ресурсу: https://youscan.io/ru/blog/smart-alerts/
7. Microsoft Azure [Електронний ресурс]. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/azure/time-series-insights/time-series-insights-overview
8. HCTSA [Електронний ресурс]. – Режим доступу до ресурсу: https://github.com/benfulcher/hctsa
9. Moving Average [Електронний ресурс]. - Режим доступу до ресурсу: https://www.investopedia.com/terms/m/movingaverage.asp
10. Python [Електронний ресурс]. - Режим доступу до ресурсу: https://docs.python.org/3/
11. NumPy [Електронний ресурс]. – Режим доступу до ресурсу: https://numpy.org/doc/stable/reference/

12. Docker [Електронний ресурс]. - Режим доступу до ресурсу: https://aws.amazon.com/docker/?nc1=h_ls
13. FastAPI [Електронний ресурс]. - Режим доступу до ресурсу: https://fastapi.tiangolo.com/
14. SciPy [Електронний ресурс]. - Режим доступу до ресурсу: https://www.scipy.org/docs.html
15. Pandas [Електронний ресурс]. - Режим доступу до ресурсу: https://pandas.pydata.org/docs/
16. Git [Електронний ресурс]. - Режим доступу до ресурсу: https://git-scm.com/doc
17. PyCharm [Електронний ресурс]. - Режим доступу до ресурсу: https://www.jetbrains.com/help/pycharm
18. Docker Compose [Електронний ресурс]. - Режим доступу до ресурсу: https://docs.docker.com/compose/
19. Microservice architecture [Електронний ресурс]. – Режим доступу до ресурсу: https://medium.com/@IvanZmerzlyi/microservice-architecture-f8a382291ff4
20. OpenAPI [Електронний ресурс]. - Режим доступу до ресурсу: https://swagger.io/docs/specification/about/
21. GIL [Електронний ресурс]. - Режим доступу до ресурсу: https://realpython.com/python-gil/